Welcome

Bill Bartlett

billb.503@gmail.com

503.504.5644

fobfuscate
Be private in public

2FA - Why?

- Avoid Moral Hazard
  - *"Moral hazard is a situation in which one party gets involved in a risky event knowing that it is protected against the risk and the other party will incur the cost."*
  - Provide Opt-In Choices
- Avoid Legal Hazard
  - Lawsuits from parties affected by data breach
- Cyber Insurance
  - Lower premiums
  - Does nothing for those who suffer Identity Theft
- Current Technology
  - OSS / easy
  - BYOD / inexpensive

2FA – How?

- 2FA: Time-Based OTP
  - Google Authenticator App

- 2FA: One-Time Password (OTP)
  - Yubico

- 2FA: FIDO Universal 2nd Factor (U2F)
  - PKI Digital Signature
- 2FA: CLEF
  - PKI Digital Signature

fobfuscate
Be private in public

2FA - Comparison

2FA Possession

- User Experience UX
- Security
- Implementation
- Cost

2FA Biometric

- Not ready for prime time
- Analog to Digital conversion
- Slow algorithms
- Accuracy
  - False negatives
  - False positives
  - Fuzzy logic
- Sensors not universally ready
- Costly
- User resistance to biometric databases

Where we have been

fobfuscate
Be private in public

Passwords

- Security by Knowledge
  - Complexity
    - 1 number
    - 1 non-alpha
    - Mixed case
    - Minimum length
  - Aging
    - Renew every x days
  - No reuse
- Convenience
  - Memory
  - Password Reset via eMail

Password Insecurity

- Static values
- Autofill
  - Password mangers
  - Browser cache
- User habits
  - Reuse
  - Written lists
  - Patterns
  - Password Fatigue
  - Shoulder surfing
- Attacks
  - Hashing rainbow tables
  - Cross site hacks
  - Social engineering

Password Implementation

- Implementation
  - Clear text
    - Bad dog!
  - Encrypt/Decrypt
    - Secure Key management
  - Hash
  - **Hash with Salt**
- Best practice
  1. Verify new password (2x)
  2. Hash(Password + UUID)
  3. Throttling retries after 3x
- User Provision
  - PinHash VARCHAR(64)
  - UUID VARCHAR(36)

Password Implementation

Register

```
// Save the PINhash
dbUser.PINhash = Hash.GetHash(model.PasswordNew + Hash.GetSalt(dbUser.UUID),
                    Hash.HashType.SHA256);
dbUser.UpdatePassword();
```

Verify

```
// Validate the Pin
if (!Hash.CheckHash(model.Password + Hash.GetSalt(dbUser.UUID),
                    dbUser.PINhash,
                    Hash.HashType.SHA256))
{...handle fail...}
```

fobfuscate
Be private in public

Where we are going

fobfuscate
Be private in public

Google TOTP

- Security by Possession
- BYOD
- Authenticator Mobile App
  - IOS, Android
  - Offline
  - $0.00
- Simple registration
  - Scan QR code
- Awkward verification
  - Read 6 digit code on mobile device
  - Enter 6 digit code on login page

fobfuscate
Be private in public

Google TOTP Security

- 80 bit symmetrical user key
  - Mobile Device
  - Web server database
    - Encryption needed
- Challenge/Response Algorithm
  - # Time segments (30 second increments)
  - Hashed, Truncated, Digitized
- Risks
  - Predictable Challenge sequence
  - Clock drift
    - Verify 3 time offset values (-1, 0, +1)
  - The keys must be protected
  - If the master key is compromised, the whole database is compromised

Google TOTP Implementation

- Register
  - Generate random key (80 bits)
  - Provision for user
  - Display QR bar code
  - App scans the QR code
- Verify
  - Verify TOTP Enabled for user
  - Collect TOTP1 from user via keyboard
  - Retrieve key from Provision
  - Generate TOTP2 on web server
  - Verify TOTP1 = TOTP2
- User Provision
  - IdUser INT (fkey)
  - Key VARCHAR(32)
  - Enabled BIT

Google TOTP Implementation

Register

```
// Get base32 value of 80 bits
model.TOTP = totp.GenerateGoogleKey();
```

Verify

```
// verify
if (totp.VerifyGoogleOTP(model.TOTP, model.OTP))
    model.Result = "Success!";
else
   model.Result = "Fail!";
```

fobfuscate
Be private in public

Yubikey OTP

- Security by Possession
- Touch interface
- Keyboard emulation
- USB
- NFC
- $30 - $80

fobfuscate
Be private in public

Yubikey OTP Security

- 12 digit serial # + 32 digit OTP
- Dynamic values
- One time usage
- Cloud Verify
- No drivers
- No batteries
- No moving parts
- No codes to remember
- Impenetrable to malware of any kind
- 128 bit encryption
- Endorsed by Google

Yubikey OTP Implementation

- Register
  - Collect OTP
  - Verify OTP with Web API
  - Provision in database
- Verify
  - Collect OTP
  - Retrieve provision
  - Verify User
  - Verify Enabled
  - Verify OTP with Web API
- Device Provision
  - IdUser INT (fkey)
  - SerialNumber VARCHAR(12)
  - Enabled BIT

Yubikey OTP Implementation

Register

```
// Verify OTP
if (!ycloud.IsVerify(model.YubicoSerialNumber)) {...handle fail...}

// Decode the serial number
string ysn = model.YubicoSerialNumber.ToUpper().Substring(0, 12);
ysn = ycloud.ToDecimal(ycloud.ToHex(ysn)).ToString();

// Provision
...
dbUserToken.Insert();
```

Verify

```
// Verify OTP
if (!ycloud.IsVerify(model.YubicoSerialNumber)) {...handle fail...}

// Decode the serial#
YubicoCloud ycloud = new YubicoCloud();
string ysn = model.YubicoSerialNumber.ToUpper().Substring(0, 12);
ysn = ycloud.ToDecimal(ycloud.ToHex(ysn)).ToString();

// Retrieve the User/Token
dbUserToken.SerialNumber = ysn;
dbUserToken.Find();
```

fobfuscate
Be private in public

FIDO U2F

- Security by Possession
- Any U2F compliant device
- USB
- Touch interface
- Challenge/Response
- PKI Private/Public key pairs
- $10 - $20
- Register
  - User provides a tag/name for the device
  - Touch the device to mint a PKI key pair
- Verify
  - User selects the device from a list
  - Touch the device to verify a digital signature

fobfuscate
Be private in public

FIDO U2F Security

- Public Key Infrastructure
  - Private/Public key pairs
  - Elliptical Curve Digital Signature Algorithm (ECDSA)
  - Private key derived in the device from a key handle
  - Public key stored in the web server database
- Challenge/Response Algorithm
  - Random challenge value is generated on the web server
  - Challenge value is signed by the private key on the device
  - The private key never leaves the device
  - The signature (~70 bytes) is returned to the web server
  - The public key is retrieved to verify the signature
  - Keys are tied to a domain
- Universal
  - Users can have multiple devices
  - Same device can be used on multiple domains
- Limitation
  - Google Chrome version 38+ with U2F extension

FIDO U2F Implementation

- Register
  - Assemble the Registration data
    - Random challenge
    - Domain ID
    - Version
  - JS: window.u2f.register(registration)
  - Retrieve Client Data
    - Challenge
    - Origin
    - Type
  - Retrieve Registration Data
    - Public Key
    - Key Handle
    - X509 Certificate
    - Digital Signature
  - Verify Digital Signature
    - X509 Public Key
  - Provision device

- Device Provision
  - IdUser INT (fkey)
  - Device name VARCHAR(32)
  - Key handle VARBINARY(128)
  - Public key VARBINARY(128)
  - Enabled BIT

FIDO U2F Implementation

- Verify Digital Signature
  - Assemble the Sign data
    - Random challenge
    - Domain ID
    - Version
    - Key handle
  - JS: window.u2f.sign(signdata)
  - Retrieve Client Data
    - Challenge
    - Origin
    - Type
  - Retrieve Signature Data
    - User Present
    - Counter
    - Digital Signature
  - Verify Digital Signature
    - With provisioned Public Key
  - Verify Counter

FIDO U2F Implementation

Register

```
window.u2f.register([{
            challenge: "@Model.Challenge",
            version: "@Model.Version",
            appId: "@Model.AppId"
        }], [], function (data) {...}

registerU2F.ParseClientData();
registerU2F.ParseRegistrationData();
registerU2F.CheckSignature();

if (registerU2F.ErrorFree) {...handle success...}  else  {...handle failure...}
```

Verify

```
window.u2f.sign([{
            challenge: "@Model.Challenge",
            version: "@Model.Version",
            appId: "@Model.AppId",
            keyHandle: "@Model.KeyHandle"
        }], function (data) {...}

signU2F.ParseClientData();
signU2F.ParseSignatureData();
signU2F.CheckSignature();

if (signU2F.ErrorFree)  {...handle success...}  else   {... handle failure...}
```

fobfuscate
Be private in public

CLEF PKI

- Security by Possession
- Free SmartPhone App
- PKI Private/Public key pairs
  - Private key in your phone
  - Public key at Clef.com
- Register
  - Install SmartPhone App
  - Register with GetClef.com
- Verify
  - Friction-less 2FA from the future

fobfuscate
Be private in public

CLEF PKI Security

- Public Key Infrastructure
  - Private/Public key pairs
  - 2048 bit PKI
  - Private key stored on the user SmartPhone
  - Public key stored at Clef
  - Out-of-band authentication is extremely secure
  - No Security credentials are stored on your website
- Challenge/Response Digital Signature
  - Receive barcode challenge from Clef via SmartPhone camera
  - Clef app signs the challenge with Private key
  - SmartPhone delivers digital signature out of band to Clef
  - Clef verifies the signature with Public key
  - Clef posts a token to your website
  - You post the token back to Clef and receive a user ID#
  - You verify the user ID# is registered to the user logging in

CLEF PKI Implementation

- Register
  - SmartPhone is registered with GetClef.com
  - User registers with your website
  - Digital Signature authentication occurs with the Clef wave barcode
  - A token is received from Clef
  - The token is posted to Clef
  - A User ID# is received from Clef
- Device Provision
  - IdUser INT (fkey)

CLEF PKI Implementation

```
<div class="form-group form-horizontal">
    @Html.Label("Clef Authentication", htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
      <script type="text/javascript"
            class="clef-button"
            src="@Model.ClefURLJS"
            data-state="@Model.ClefDataState"
            data-app-id="@Model.ClefAppId"
            data-redirect-url="@Model.ClefURLCallback"
            data-color="blue"
            data-style="button"
            data-type="register">
      </script>
    </div>
 </div>
```

Register

CLEF PKI Implementation

```
<div class="form-group form-horizontal">
    @Html.Label("Clef Authentication", htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-9">
      @if (Model.success)
      {
        <script type="text/javascript"
              class="clef-button"
              src="@Model.ClefURLJS"
              data-state="@Model.ClefDataState"
              data-app-id="@Model.ClefAppId"
              data-redirect-url="@Model.ClefURLCallback"
              data-color="blue"
              data-style="button"
              data-type="login">
        </script>
      }
      else
      {
        @TempData["Message"]
      }
    </div>
  </div>
```

Verify

fobfuscate
Be private in public

2FA Summary

User Experience	Google TOTP	Yubico OTP	FIDO U2F	Clef.com
Form Factor	Mobile device	Hardware token	Hardware token	Mobile device
User Interface	App	USB / NFC	USB	App
User Action	6 digit transcribe	Touch activated	Touch activated	Point camera at the computer
Algorithm	Symmetric Encryption	Symmetric Encryption	PKI Digital Signature	PKI Digital Signature

fobfuscate
Be private in public

2FA Summary

Security	Google TOTP	Yubico OTP	FIDO U2F	CLef
Algorithm	Public algorithm	Vendor specific algorithm	ECDSA	DSA
Challenge Value	Time period (predictable)	Random Counters	Random Challenge	Vendor
Method	Calculate TOTP on web server and compare to user value	Cloud Verification Web API	(a) Digital Signature verify on web server (b) Verify counters are incrementing	(a) Out of band challenge and response (b) Async result notification
Weakness	90 second window for replay, Clock drift	No replay	Requires Chrome browser	

fobfuscate
Be private in public

2FA Summary

Integration	Google TOTP	Yubico OTP	FIDO U2F	CLef
Key Management	Keys are vulnerable in web database unless encrypted	Keys are stored in Yubico cloud	Public keys stored on web database, Private keys never leave device	Public key stored at Clef Private stored in SmartPhone
Complexity	Easy	Moderate	Difficult	Difficult
Browser	Any browser	Any browser	FIDO extension required (Chrome 38)	Any browser
BYOD	$0 (SmartPhone)	$30 - $80	$10 - $20	$0 (SmartPhone)
Server Cost	$0	$0	$0	Volume pricing

2FA Summary

- Best Practices
  - Self Service
    - Opt-In
    - Enroll
    - Repudiate
  - User Notification
    - Email
    - SMS

DEMO

Give Identity Theft the Finger